Vision-based People Counting System

The Undergraduate Research Opportunities Scheme (UROS)

Author: David Whitehead

Supervisor: Dr Grzegorz Cielniak

Project Date: 14/06/2011 – 22/08/2011

University of Lincoln

Contents

1. Introduction

Being able to count the number of people entering or leaving a place can be very important. This is because it gives the ability to do tasks based on the amount of people that are within an area. For example, to divert the pedestrian traffic to another location or to not allow anyone else into a football stadium, as there are is no room left etc. A people counting system could even be used to help local events, such as the Lincoln Christmas Market, as it would provide the knowledge to know where to divert people if certain areas of the market are overcrowded.

The project which will be discussed will be researching and developing a fast and effective people counting system. The project will be 10 weeks long and within this time period a basic solution and advanced solution will be completed. The first couple of weeks the project will be focused on the basic counting system, and then the rest of the time will be given to the advanced system. The report is structured by showing the research first then the hardware and software setup of the project and the different stages of the people counting system.

1.1. Research

Research has been done to help finding the most effective methods of counting the number of people going in and out of a specific scene. There are many different types of people counting devices used from vision-based counting to a tally counter or even infrared beams. It is clear that tally counting is not as effective because it is possible to miss a count if people are walking too fast or there are too many people together, as this would affect the final result of finding out how many people there are in total.

It is also clear that devices such as infrared beams have limitation to where they can be installed and how accurate the people counting method is depending on this, as stated by NationalTrail (2006), “*Different sensors work in different ways. This affects what they can count, how accurate they are and where they can be installed.*“ for example *“A beam is broken or reflected - these can only be installed on narrow paths*”.

Vision-based counting systems use some form of camera to be able to count people walking past. Vision-based counting systems use image manipulation techniques to be able to distinguish different elements in a scene. Some of the basic counting systems only use a normal mounted camera, whereas others use a more sophisticated type of camera such as a stereo camera. The fact that stereo cameras are becoming cheaper seems to enhance the development and research that is being done on them to be able to count people. This is because it not only gives a general picture but also supplies the depth information from a scene, this is also stated by Muñoz-Salinas et al (2005), “*Although the topic has been extensively explored using a single camera, the availability and low price of new commercial stereo cameras makes them an attractive sensor to develop more sophisticated applications that take advantage of depth information.”*

Although using a stereo camera to be able to detect people based on the extra depth information is a very efficient way of counting people, it also means that more computation is being done as more information is being processed from the camera, due to not only capturing one single image but two images with slightly different phases to be able to simulate the human Binocular Vision. This is supported by Caron and Eynard (2011), *“A stereo (or stereoscopic) rig is a device with two or more cameras that makes it possible to simulate human binocular vision and its ability to capture 3D images”*

Muñoz-Salinas et al (2005) also stated that this is not only taking two images, but then it has to be put through a process to find the points throughout the images that are the identical, this is also known as disparity image, A stereo camera *“can capture two images from slightly different positions (stereo pair) that are transferred to the computer to calculate a disparity image Id containing the points matched in both images.”*

Another interesting method that has been previously used with vision-based counting is to be able to track and label objects within a scene. These types of methods are useful as the objects within a scene can be classified as people and therefore be counted as such. Once the objects are being labelled in a frame it then gives the ability to be able to track them from one frame to another, as stated by Kim et al (2002), *“In the object tracking algorithm, the center point of”* a pedestrian, *“verified as a pedestrian in the previous image is predicted in the current image by analysing the past velocity information of the moving object”*.

Most vision-based methods from the basic camera to a sophisticated one use a similar approach in respect to how the camera is mounted. The research shows that cameras are generally mounted above a door to give a slanted or top down view for most of the people counting systems, this is also stated by Muñoz-Salinas et al(2005), *“Several authors have mounted their cameras in the ceiling to perform people detection in Ambient Intelligence domains”* whereas *“Others have used slating cameras in overhead”*.

1.2. Hardware and Software Setup

The hardware setup for the project consists of a camera mounted above a door giving a top down view. This camera position should help to support the people counting system as it will help to stop people overlapping each other. The camera which is being used is a “USB2 Webcam Live WB-5400”, which is a 4 megapixel camera. The camera resolution is 640 by 480 pixels and the picture quality is standard.

See Figure 1 for a picture that shows the mounted camera.

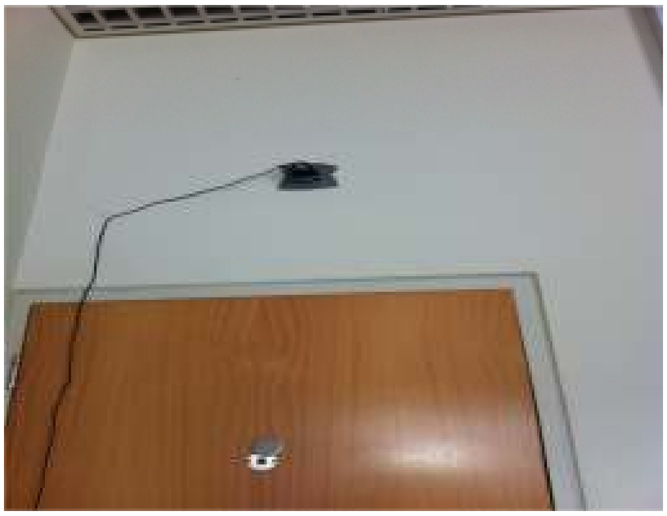

Figure 1. Camera mounted above the door.

The software which is being used is the open source image processing library OpenCV (OpenCV, 2011), this will give the ability to be able to get the feed from the webcam as well as be able to perform image processing functions on it. The library is a source of many different processing functions that help to distinguish the objects in the scene from the background etc. The OpenCV library also has an easy to use Graphical User interface, which will be helpful in implementing and testing the algorithms.

C++ is the main language which the OpenCV library is running on through this project. C++ was chosen as it’s the language which is more efficient in terms of accessing memory and using pointers than other languages. C++ was also chosen as it helps to make the algorithms more portable to other operating systems as it is a more generic language.

2. Basic Counting System

2.1. Introduction

The basic people counting algorithm is based on pixel change detection, meaning the algorithm uses the changes in pixel colour to count people. The algorithm separates the foreground and background pixels by using background subtraction and then determines how many pixels there are when a person is walking past the camera. This process starts by getting the difference between the current frame and the background frame to be able to obtain only the foreground. Then the difference is put through thresholding, thresholding is where an image is converted based on a parameter to black and white, which means in this case it makes the foreground white and the background black. After obtaining the thresholded image the next process counts how many white pixels there are in the image when a person enters and leaves the screen. This will give the ability to determine how many white pixels count as a person, thus allowing a limit to be set for how many white pixels actually count as a person. See Figure 2 for a diagram of the process order.

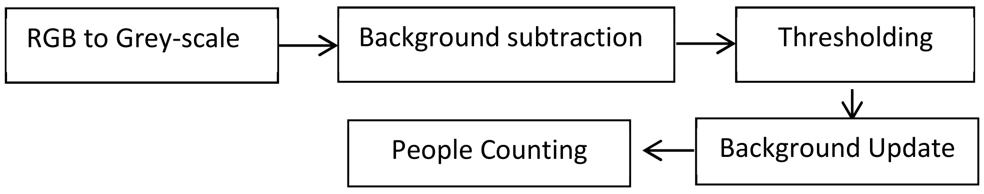

Figure 2. Basic counting system diagram.

2.2. Grey-scale

The first process in the basic algorithm is to convert the current RGB source input into a grey scale image. The algorithm uses one channel as it is quicker and easier to do calculations on then it is to do on three channels. This is because the three channels are using the red, green and blue rather than just a light intensity from 0-255. When an image gets converted into grey scale it is simply given one channel from 0 to 255 and depending on what the value is depends on what shade of white it is. For example 0 would be black and 255 would be white. See Figure 3 and 4 for a normal RGB image which is then converted into grey-scale.
The formula stated by Bradski and Kaehler(2008) is as followed,
“Y = (0.299)R +(0.587)G +(0.114)B ”, where R is red channel, G is green channel, B is blue channel and Y is for the resulting gray-scale value.

Figure 3. RGB image.

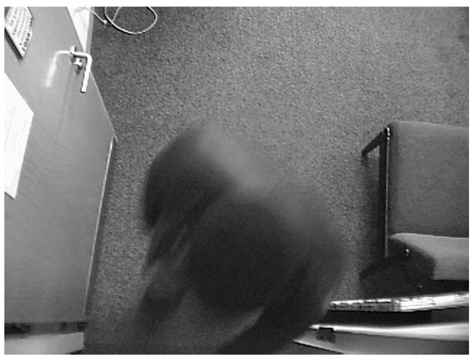

Figure 4. Converted grey-scale image.

2.3. Subtraction

Now that the source has been converted to grey-scale it is now possible to do background subtraction. Background Subtraction is where a difference image is obtained by comparing the current frame to the background frame in order to determine the “difference” of the two images. The “difference” after the background has been removed is also known as the motion. See Figure 5 for an example of a difference image created from the background frame and current frame. The formula is as followed, $l(x,y) = (a(x,y) – b(x,y))$ where a is the background image, b is the current frame and l is the resulting image.

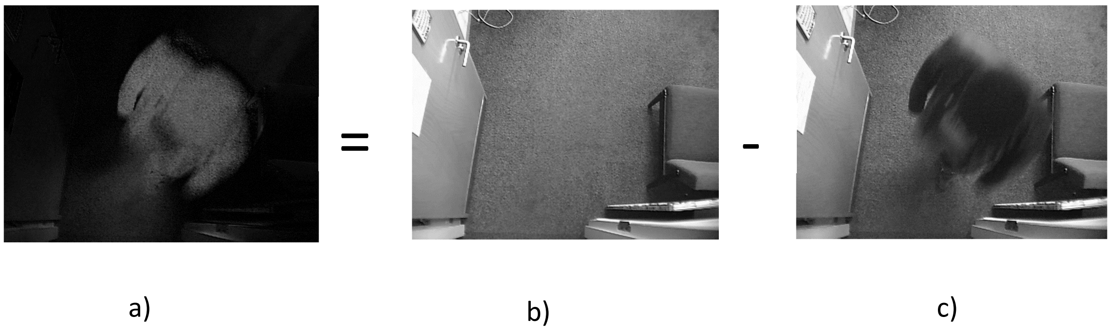

Figure 5. The result of background subtraction: a) the resulting image, b) background image, c) current image.

2.4. Thresholding

Once the background subtraction has been completed it is then possible to determine what the foreground is. To achieve this result the image difference from the background subtraction, is put through thresholding. Thresholding is where all the pixels in an image with an intensity higher than the specified threshold value, are considered to be the foreground(white) and the rest is the background (black), which means thresholding converts an image to black and white based on the threshold value supplied. For example, if 200 was the value, then anything below 200 in the image colour would be counted as black “0” and anything in-between or equal to it would be counted as white “1”. By using thresholding it becomes possible to detect people by filtering out the unnecessary parts of the image, it also ensures that problems such as light changes and flickers are not counted within the detection, as it will probably be below the threshold which would ensure it will be set to 0 “black”. See Figure 6 for a resulting image after the thresholding operation. The formula is as followed where T is the resulting image, A is the current input frame and threshold is the threshold value supplied.

$$T(x,y) = \begin{cases} 1 \text{ if } a(x,y) >= \text{threshold} \\ 0 \text{ if } a(x,y) < \text{threshold} \end{cases}$$

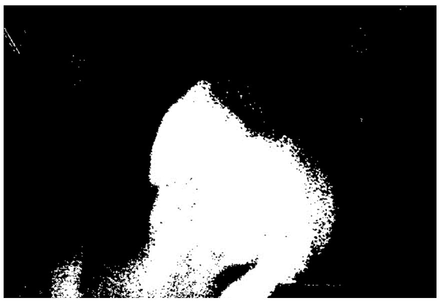

Figure 6. Thresholded image with a minimal value of 30.

2.5. Background update

Once the background and foreground have been obtained it is then important to allow the updating of the background. This is important as if something new gets added to the scene or the camera is moved, the background needs to update itself so that it’s not constantly out of sync. The formula that is going to be used is as follows,

$\mathrm{acc}(x, y) = (1-\alpha)\cdot \mathrm{acc}(x, y)+\alpha \cdot \mathrm{image}(x, y) \text{ if } \mathrm{mask}(x, y) \neq 0$

The running average is used in order to fill in the average intensity value within two images. The speed at which this is done can vary from the values being supplied to depending on if a mask image is also being supplied or not. The running average in this algorithm is being used twice to be able to update the background slowly and the foreground reasonable quickly. This will help to ensure that the camera and background can be changed “moved” at run time and will adapt to situations even if it is slowly. It will also ensure that the updating of the foreground is done fast, this will ensure the foreground keeps completely up to date for the detection algorithm. See Figures 7, 8 and 9 for the resulting images.

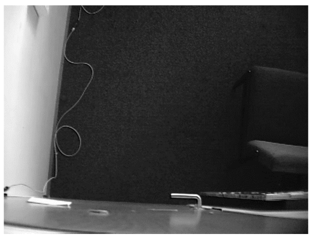
Figure 7. Initial background.

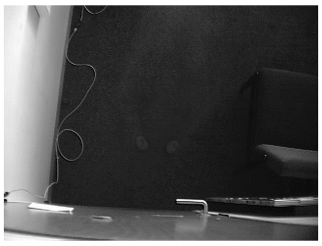
Figure 8. The update process starting.

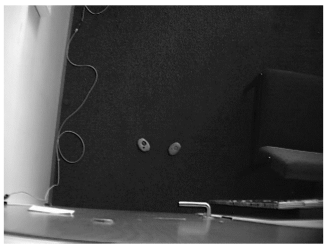
Figure 9. The objects finished updating.

As you can see in Figure 8 the background is normal. Then in Figure 9 a person comes and puts two balls onto the floor. Then the resulting image of the background updating process is Figure 10, where the objects have now been added to the background.

2.6. People Counting

The detecting algorithm is based on the white pixels from the threshold frame. In order to determine how many white pixels are counted as a person, data has been collected from many different scenarios and a summary was produced. Some of the general scenarios in which the data came from were video files that had been recorded of a person or multiple people walking in and out of a room, some of the scenarios also included data with a person wearing a coat and bag etc. With this data it has made it possible to determine how many white pixels are counted as a person, so that a threshold limit could be set for the algorithm. This was done by examining the summary produced and looking at the ground truth and pixel spike rates in each scenario and finding a general basis of what counted as a person.

See Figures 10, 11, 12 and 13 for some of the graphs used where the ground truth is 5000 per person. All the graphs show the foreground pixel count as blue and ground truth as red for each sequence.

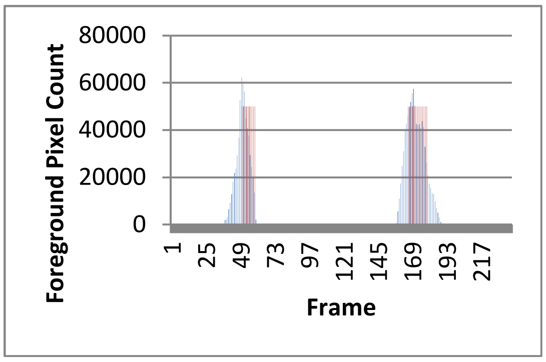

Figure 10. A sequence with a single person.

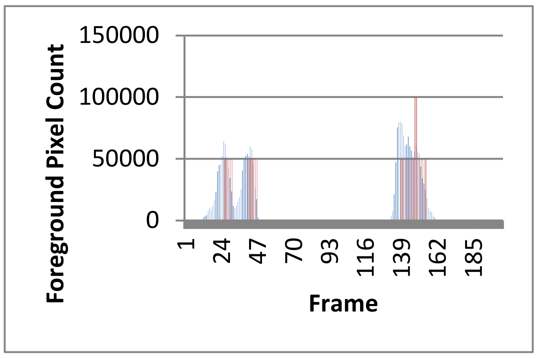

Figure 11. A sequence with two people.

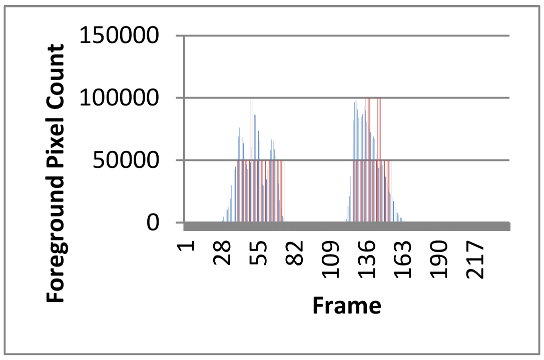

Figure 12. A sequence with three people.

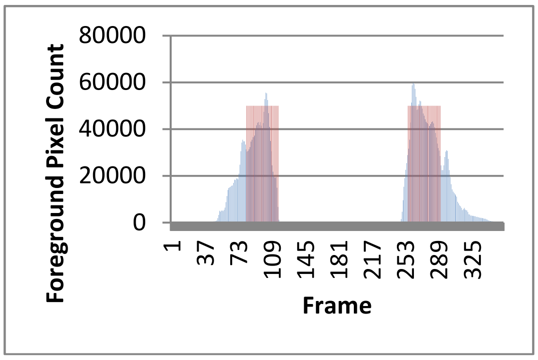

Figure 13. A sequence with a single person walking slowly.

2.7. Counting Algorithm

A general assumption which could be made could be that anything above a specified threshold is defined as a person and can be counted as one. Another assumption that could be made based on the data retrieved is the possibility of setting pixel base lines, for example anything above 60000 at a time is a person, but anything above 85000 at a time is two people. This idea assumes that it is being based on a person being average build, height and width etc.

2.8. Averaging the pixels

In the white pixel graphs above the general pixels show a spike when a person enters or leaves the room but the spike fluctuates a lot. To help to solve this issue an averaging algorithm has been created to help when a person enters or leaves, ensuring the spike stays the same and does not fluctuate much within this process.
The averaging method works by using a variable to store up to N values, which are being updated every frame based on the total white pixels. If the variable already holds the maximum values, then the first value in is popped out and the new white pixel count for that frame is inserted at the back “Similar to a queue effect”. Once the variable has been updated for that frame then all the values are averaged. The output is then compared to the person threshold value to see if a person is entering or leaving the room based on the pixel count. By using this method the white pixel threshold value per person has had to be changed, as the threshold value per person has dropped lower because of the averaging. Although the averaging method has helped to ensure that the number of false triggers as a person being counted has improved dramatically due to the pixel spikes being more stable.

See Figures 14 and 15 for some two of the graphs showing the improved spikes using the averaging method. Both graphs show the foreground pixel count as blue and the averaging count as red for each sequence.

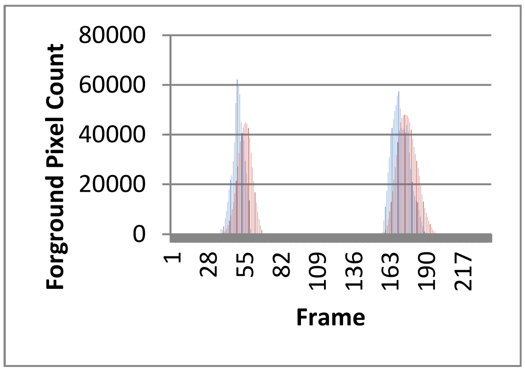

Figure 14. A sequence with a single person.

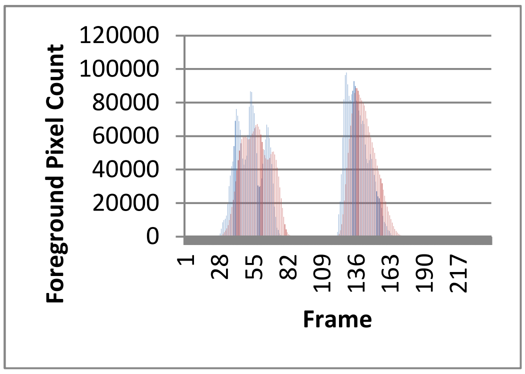

Figure 15. A sequence with three people.

2.9. Direction Detection

As the camera in this scene is mounted above a door looking down, to detect whether people are leaving or coming in, the camera frames have been halved, so that once the white pixel threshold per person is going below the set amount “a person is going off the screen from being on it”, the method calculates whether the amount of white pixels at that time are higher on the top or bottom of the screen. This then gives us the ability to create a basic exit and entrance algorithm to be able to count the person as leaving or entering the room. See Figure 16 for an image of the direction detection system.

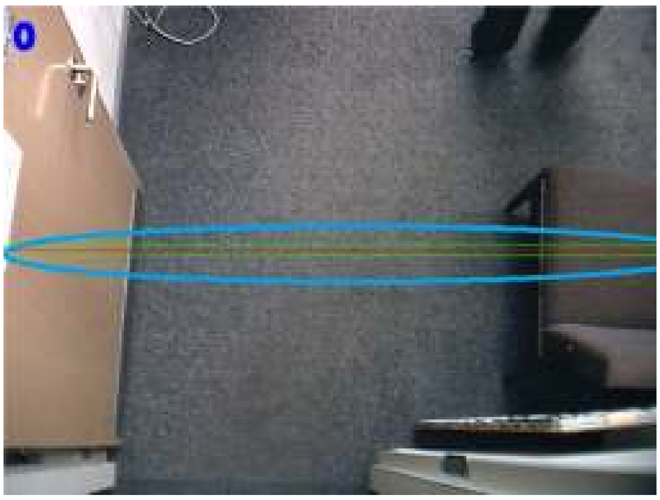

Figure 16. Direction Detection Image.

2.10. Strengths and Limitations

The main advantage of this algorithm is its fast ability to count people and determine whether or not the person is going in or out the room. The algorithm is quick as it is not based on trying to follow a person from frame to frame, but instead the algorithm is simply checking the amount of white pixels per frame. As this is natural data it means the amount of computation needed is a lot smaller than an algorithm which tries to detect a person and follow them from one frame to another.

The general disadvantage to using standalone pixel counting in this case is the fact that it is not possible to determine accurately if multiple people are in the scene. This is due to people clustering together and the white pixel count fluctuating so much that it’s not possible to get an accurate threshold to say when multiple people are involved.

2.11. Evaluation

The white pixel detection algorithm has difficulties in detecting more than one person at a time, even with the averaging method which was created to help reduce the spike fluctuations in the white pixel count, it is still problematic.

The overall conclusion is that this type of algorithm is a good and fast single person counting algorithm, it can do the general basic i.e. detecting single people one after the other, updating to a different location, allowing for changes in threshold data, determining whether the person is leaving or entered the room etc. But its ability to count multiple people based on the white pixels count, is inefficient as the white pixel count fluctuates in value so much that when there is more than one person at a time, it makes it very difficult to be able to setup a stable and accurate threshold value for counting multiple people in the scene.

2.12. Computation Time

Below is a list of computation times to help show how fast each step within the basic counting algorithm is taking. As you can see from the results in Figure 17 the majority of the time is spent on the conversion to grey scale and updating the background. The times were taken after 5 runs of 1000 each time to get a more accurate speed result.

Testing Functions	**Time [ms]**
Converting image to grey scale	2.3 +/- 0.03
Subtraction	0.6 +/- 0.05
Threshold	0.2 +/- 0.02
Running Average(background update)	3.5 +/- 0.08
Running Average(foreground update)	0.8 +/- 0.04

Figure 17. The computation time results for the basic counting algorithm.

3. Advanced Counting System

3.1. Introduction

The advanced people counting algorithm is based on motion pixel accumulation. The algorithm first obtains the motion information from the scene by subtracting the current frame from the previous one. After the motion is obtained the algorithm uses the motion to determine how much is being accumulated by people going past the specific lines. Depending on the threshold values set for them lines depends on if it will count a person or not. See Figure 18 for a diagram of the process order.

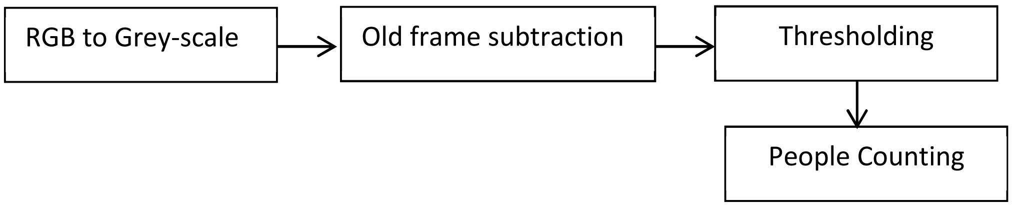

Figure 18. Advanced counting system diagram.

3.2. Motion Detection

The motion is obtained by subtracting the current grey scale image from the previous frame. This means that the output is pure motion as everything that is the same in both frames is subtracted and is zero. The motion is then put through thresholding to ensure most of the noise from light etc is cut out and the only motion obtained after thresholding is someone moving “this means only white will be where something is moving and everything else will be black”. The motion is an important part of the algorithm as it ensures that when the accumulation method is being used, if the person stops on the line it will not constantly keep accumulating ,because as soon as the person stops moving and stays still, the pixels will turn to black “0” as there is no motion no the scene.

See Figure 19 and 20 for images of the motion before and after thresholding.

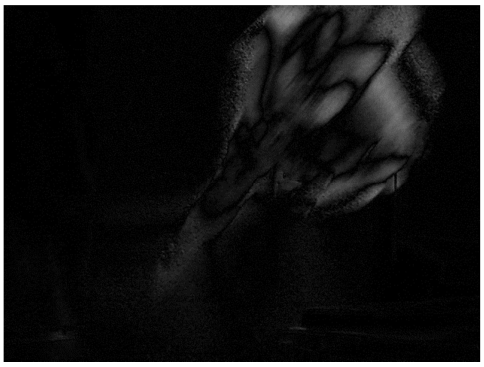

Figure 19. Motion of a person moving

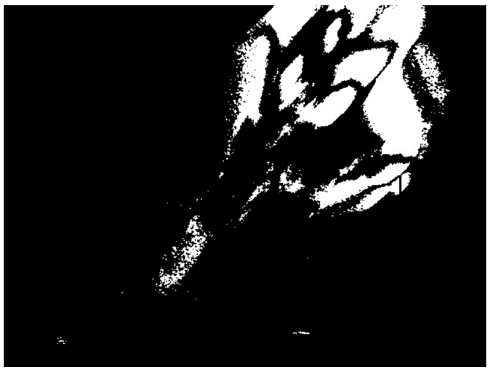

Figure 20. Image after the motion has been thresholded

3.3. People Counting

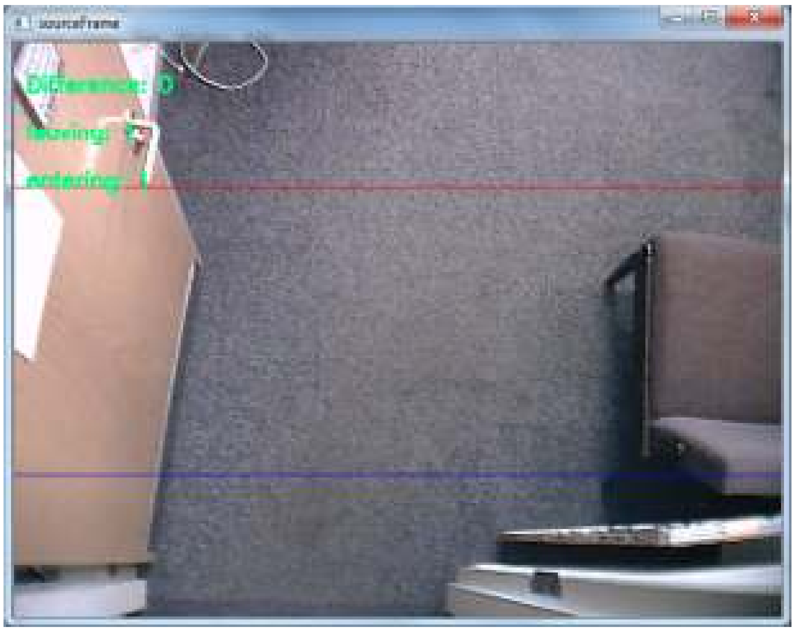

Figure 21. Upper and bottom lines used for people counting.

The algorithm works by using two lines within the scene as you can see from Figure 21. It uses the lines by constantly counting the motion pixels that run along these two lines. Each of the two lines has a minimal and maximum threshold value; these values are used to count people. The values determine that anything above the minimal and not above the maximum is a person but if it goes over the maximum value, the counter is reset to zero and if it reaches the minimal value again then its counted as another person. For example, if it detects that the white pixels count is above the minimal threshold value for that specific line, it counts it as a person and starts to accumulate the total of the white pixel counts for every frame that it’s above the minimal value. This is done to make sure that if the accumulated total goes past the maximum threshold value, it resets the counter to zero to start the counting process again for the next person. Otherwise the reset flag is only used when the white pixel count is zero which means there is no motion on the line.

When the motion pixel count hits the minimal threshold value, the algorithm checks to see whether or not the person has crossed the bottom line before crossing the top line, this gives the ability to be able to determine whether or not the person is leaving the room or entering it. This is done via using a flag that counts the bottom line. See Figure 22 for a detailed diagram of the process.

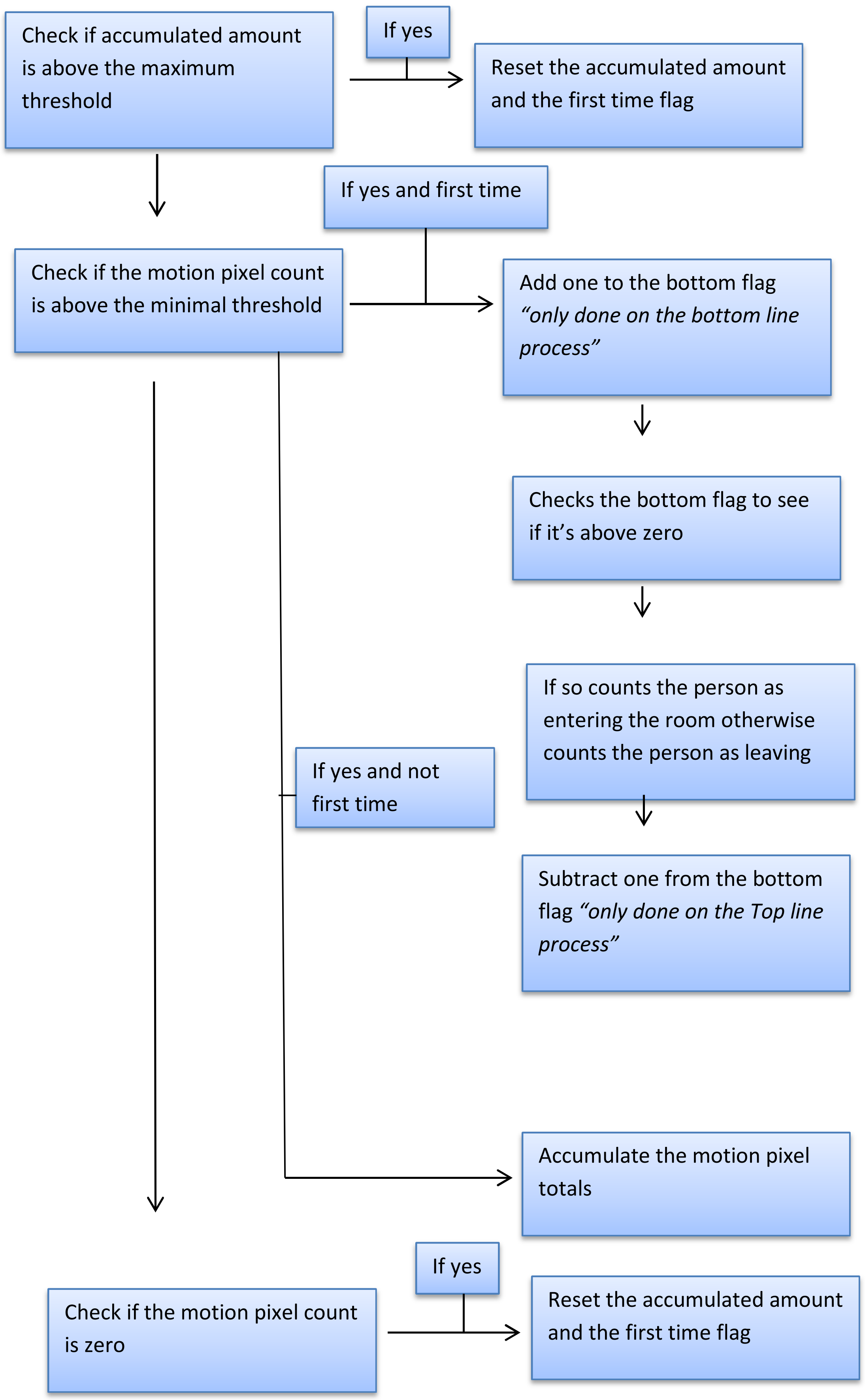
Check if accumulated amount is above the maximum threshold
If yes
Reset the accumulated amount and the first time flag
If yes and first time
Check if the motion pixel count is above the minimal threshold
Add one to the bottom flag *“only done on the bottom line process”*
Checks the bottom flag to see if it’s above zero
If so counts the person as entering the room otherwise counts the person as leaving
If yes and not first time
Subtract one from the bottom flag *“only done on the Top line process”*
Accumulate the motion pixel totals
If yes
Check if the motion pixel count is zero
Reset the accumulated amount and the first time flag

Figure 22. People counting diagram.

3.4. Threshold values

To be able to set the line threshold values, data has been collected and put into graphs using the same scenarios that were used in the basic counting algorithm. From the data collected a summary was made to help give the maximum and minimal values for both the top and bottom lines in the algorithm. The top lines maximum value has been set to 5500 and the minimal has been set to 1000 as well as the bottom lines maximum value has been set to 2500 and the minimal has been set to 400. The threshold values where calculated by looking at the data produced in the graphs and finding a stable minimal and maximum motion pixel count for a single person. See Figures 23-26 for images of some of the graphs. All the graphs below show the line motion pixel count as blue for each sequence.

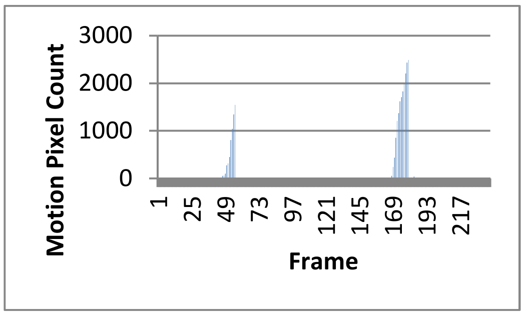

Figure 23. Top line sequence with a single person.

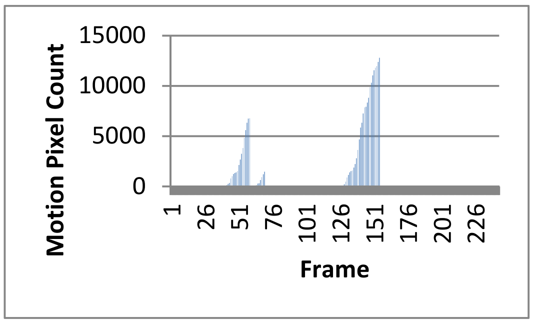

Figure 24. Top line sequence with three people.

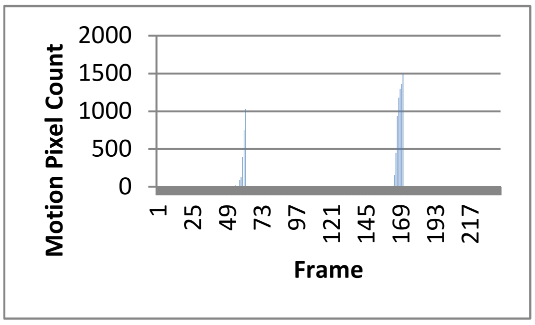

Figure 25. Bottom line sequence with a single person.

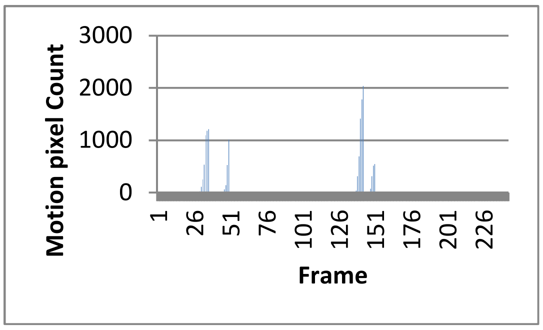

Figure 26. Bottom line sequence with three people.

3.5. Strengths and Limitations

One of the biggest advantages that this algorithm has is its ability to count multiple people quickly. The algorithm does not run detection from a frame to frame basis which ensures that the algorithm runs smoothly. The algorithm runs even quicker than the basic counting algorithm, as it is only checking two entire lines for the pixel counting rather than the entire image.

The main disadvantage to this algorithm is the impact of someone walking half way through the scene and then turning round; as this algorithm assumes that everyone is walking all the way through the scene. If a person does this it will mean the flag ends up false and the count is inaccurate. Another disadvantage to using this algorithm is the fact that the line thresholds will have to be re adjusted and manually updated every time the scene changes to another location.

3.6. Evaluation

The advanced counting algorithm is very efficient in speed and computation compared to the basic counting system; see the advanced computation time's section for the speed calculations. The advanced counting algorithm can count multiple people going in and out of a room. It does not matter how close the people are to each other as it's not counting the separate entities in the camera feed, simply the amount of motion accumulating on the line. This means that other than getting the motion from camera feed, the computation involved in checking a single line all the time is far smaller than trying to identify individual objects within the camera feed to be able to count them.

But one of the main disadvantages to the advanced counting algorithm is the fact that its assuming everyone is walking through the entire scene and not turning back on themselves half way through. This is because the algorithm uses a decrementing flag which means if a person does not do as assumed then the system counter will not only count incorrectly, but it will decrement the flag in a way which the algorithm was not designed thus causing issues for the future counts.

4. Further Extensions

4.1. Introduction

This section presents further updates to the original advanced counting algorithm. The algorithm is now broken up into three different phases. The first phase determines the entrance and exit points via accumulating and creating an image of the path people are walking in the scene. Once the path has been defined a border checking technique is used to find the biggest line on the top and bottom of the window. The second phase calculates and learns the line threshold values for the new lines which have been detected. The third phase is to calculate like normal the amount of people walking in and out the scene. As the third phase is the same as the first version of the algorithm, only the updates to the algorithm will be discussed in this section. Parts of this algorithm are learning based, meaning it would have to be on for a long time for accurate values to be obtained. See Figure 27 for a diagram of the process order.

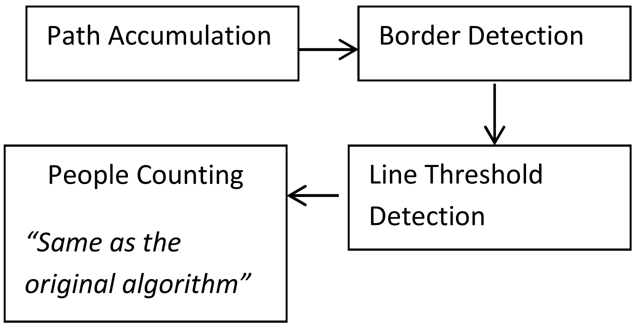

Figure 27.Further Extensions diagram.

4.2. Exit and Entry Detection

4.2.1. Path Accumulation

The accumulation algorithm works by using a 32 floating point image to store the accumulation of the motion images overtime. It uses the OpenCV add function, to be able to accumulate the motion image with the current storage image. This gives the ability to be able to see which areas are more frequently visited then others within the scene. Then the image is rescaled down to a ratio of 0 to 255. This is done by finding the highest number in the 32 floating point image and then using it in the OpenCV scale function as the maximum scaling number by doing the following formula, “255/the maximum of the 32 floating point image.” The output is then put through thresholding to ensure low parts of the accumulation are discarded. See Figure 28 and 29 for images of the resulting accumulation process.

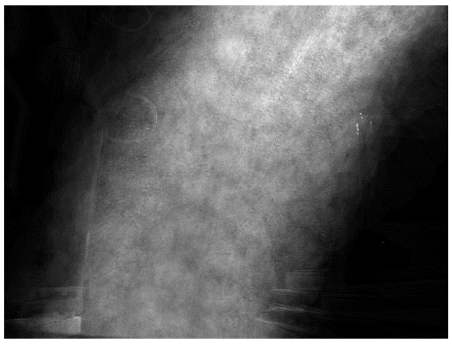
Figure 28. Scaled accumulated path.

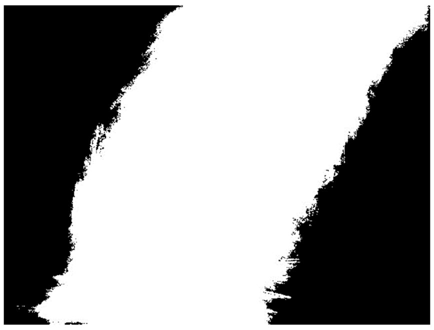
Figure 29. Thresholded path.

4.2.2. Border Detection

The border detection algorithm takes the thresholded image and checks the entire top and bottom lines within the image for the largest accumulated pixel lines. Once it finds a line that is over a certain amount of pixels connected together, it stores it into a queue. After all the checking has been done it then runs through the queue while checking the size of each line and then takes the largest of both the top and bottom lines. Next the co-ordinates of both the start and end of the lines are passed back for the next section. See Figure 30 for an image of the lines that are created from this algorithm.

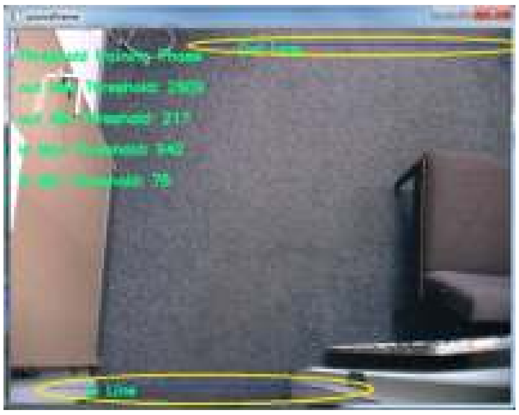
Figure 30. Border Detection Result.

4.3. Line Threshold Detection

The line threshold detection algorithm uses the lines obtained in the border detection to find the minimal and maximum values for each line. The maximum value is found by running through the entire line while counting any white pixels from the motion, if it finds a value which is higher than the current one, it replaces the value and carries on searching through the line. The minimal value is obtained by calculating 12 percent of the maximum value that is obtained for each line. See Figure 31 for an image of this.

Figure 31. Line Threshold Values.

4.4. Strengths and Limitations

The main advantage of this algorithm is its ability to not only count multiple people but the fact that it also can be put anywhere and learn the path of where people are entering and exiting, as well as the threshold values for the each of the lines. By having the new learning phases it gives the ability to ensure that the algorithm is a lot more module and portable in the different possible scenarios.

The main disadvantage to this algorithm is the fact that it still has the same problems in which the first version had in terms of counting people that are not walking all the way through the scene but are turning round.

4.5. Evaluation

The updated advanced counting algorithm is an overall bundle of all the different advantages leading up to this algorithm. It takes the ability to count multiple people from the first version and then adds in an extra two different learning phases to ensure that the algorithm is not only effective, in terms of learning its surroundings, but to ensure that it can be portable in moving to a different location.

The main problem with the algorithm is that it still lacks the ability to be able to count people which are not fully walking through the entire scene. This issue conflicts with the flags in the algorithm and sets of a bundle of problems off if it occurs.

4.6. Computation Times

Below is a list of computation times for each process of the algorithm. The results show how fast each of the different processes are. As you can see from the results in Figure 32 the main process that requires the most time is the path accumulation method. This is because the method is always accumulating which is costing a lot of computation whereas the methods, line threshold and people counting are simply checking lines rather than an entire image. As you can also see from the results in Figure 32 the lowest computation process through the entire algorithm is the thresholding method.

Testing Functions	Time [ms]
Converting image to grey scale	2.2 +/- 0.05
Subtraction	0.8 +/- 0.04
Threshold	0.1 +/- 0.02
Path Accumulation Calculation	3.8 +/- 0.03
Line Threshold Calculation	0.9 +/- 0.02
People Counting Calculation	0.7 +/- 0.02

Figure 32. The computation time results for the advanced extend algorithm.

4.7 Accuracy

The counting ratio's where taken from the output results of the algorithm running through 9 different videos, the videos vary from multiple people being in the scene to singular. The true negative will not be implemented into the ratios as when there is no one in the scenes of the videos it will always be true negative. As you can see from the results in Figure 33 the algorithm does have its flaws, the 6 percent which was false positive was on a video of a person walking half way through the scene and then back again, the assumed system does not run on this basis thus causing the false positive twice. All the other videos of people walking and in out of the scene where counted correct as you can see from the true positive results. The false negative results also show that the algorithm did not miss a person when there was one in the scene.

True Positive	False Positive	False Negative
93.94%	6.06%	0%

Figure 33. Counting ratio percentages.

5. Project Conclusion

The overall aim for the project was to be able to create a people counting algorithm that also was done in a way in which the computation required would be as low as possible. Generally the overall project has succeeded in its main aim to create a people counting algorithm, which can count multiple people and even does so with little computation. The project even expanded to be able to create a basic exit and entrance learning algorithm to help ensure the portability into different places.

The project did have some difficulties in the fact that it assumes people are walking entirely through the scene and are not stopping and turning round. Some of the other difficulties throughout the project were things such as trying to find a method of count people without relying on tracking and blob detection etc. as that would increase the computation of the algorithm.

Some of the improvements which could be changed in the future algorithms would be things like not assuming that people are walking through the entire scene and adapting the counting algorithm to check if this is occurring. Another update which could be done to improve the project is to design a more flexible entrance and exit detection system so that it could be based not only on the borders of the scene but anywhere throughout it. This would ensure that if a scene which is being monitored does not have an entry or exit point starting from the top or bottom of the image, it can still be setup.

Throughout the project I personally have gained experience with C++ and OpenCv programming. Before the project I had never used OpenCv and only knew the basics of C++, but this project has allowed me to grow on my previous knowledge and learn even more within software developing. I have also gained experience with using video editing software such as virtual Dub, which has allowed me to cut and join different video footage for testing purposes. Through the project I have also gained knowledge in manipulating images which I have never done previously. I feel I have grown in respect to what the possibilities are within programming, as I think that what I have learnt could now be used in a number of different ways. I also feel that throughout the project, the way in which I work and write has become more efficient in terms of correct language being used and problem solving.

Overall I think now that I have gained more experience with this type of programming, if I was to start the project again I would start the research in a different area, as I think motion tracking from one frame to another could have its benefits in a people counting system.

Bibliography

Muñoz-Salinas et al(2005) *People Detection and Tracking Using Stereo Vision and Color*. Page 3, 6, 8

Kim et al(2002) *Real-time Vision-based People Counting System for the Security Door.* Page 3

Caron and Eynard(2011) *Hybrid Stereoscopic Calibration.* Page 1

Bradski and Kaehler(2008) *Learning OpenCV.* Page 60, 136, 292

James Davis(2001) *Representing and recognizing Human Motion: From Motion Templates to Movement Catagories*

J.D Valke et al(2007) *People Counting in Low Density Video Sequences*

Guillem Pratx(2004) *Building Detection*

Ritter and Wilson(n.d) *Handbook of Computer Vision Algorithms in Image Algebra*

NationalTrail (2006) *People Counting Information* [online] available at http://www.nationaltrail.co.uk/uploads/people%20counter%20info%20summary.pdf [Accessed on 12th August 2011]

OpenCV (2011) *OpenCV* [online] Available at http://opencv.willowgarage.com/wiki/ [Accessed on 12th August 2011]